

Version 4, June 2015, ELAN 4.9.1

INTRODUCTION

This README provides basic information on how to build an audio, video or other media (e.g. timeseries) recognition tool that can co-operate with ELAN. Defining the interface between ELAN and audio-/video/other-recognizers is work in progress; this document describes the fourth version of the interface and highlights the changes since the previous version.

Feedback on the interface and on implementing a recognizer extension can be given on the ELAN forum: <https://tla.mpi.nl/forums/software/elan/>

Before experimenting with the demo recognizer in this package and subsequently your own recognizer it is advised to read the paragraph in the ELAN manual regarding the silence recognizer and the Recognizer panel.

CHANGES

Version 4: The RecognizerHost has one new method: `appendToReport(String)`, which is now the default way to send messages about the process to the user. The method `getSelectionPanel(String)` now returns an `AbstractSelectionPanel` object. The Recognizer interface also has one new method: `updateLocaleBundle(ResourceBundle)`, which not only informs the Recognizer of a change in the interface language but also gives access to the translated key-value pairs available in ELAN.

The Audio and Video tabs in ELAN have been merged into one Recognizer tab, listing the recognizers for all sort of media, including timeseries data. A recognizer will be informed of the linked files for its main type (e.g audio) in a call to its `setMedia(List)`. If a recognizer wishes to have the list of other media types (e.g. video, timeseries), it can get them from the recognizer host by calling `getMediaFiles(int)`.

Version 3: The composition of the graphical user interface of the recognizer panels has changed. The Selection panel that allowed to add multiple manual selections and/or complete tiers has been removed from the main panel. Likewise for the list of media files. These items have been moved to the configuration or parameter panel of the recognizer. A new panel has been developed that allows to either add custom selections, or to select a tier, or to select a file as input to the recognizer. If a recognizer does not have its own control panel this new Selections panel will be added to the UI for the proper parameters based on the metadata. If the recognizer does have its own control panel, it can obtain the new Selections panel from the recognizer Host. The host will call the `setMedia(List)` method to pass the linked media files to the control panel.

The method `getExampleSupport()` has been removed from the Recognizer interface.

It is now possible to have multiple `cmdi` files in the same folder in the extensions folder, the name `recognizer.cmdi` is no longer mandatory.

Version 2: There are two major changes in the recognizer extension mechanism since the first version. Recognizer components now have to be installed in their own directory/folder inside the extensions directory and should provide a component metadata file. In addition to the existing extension option based on the Recognizer API, which means implementing the (Java) Recognizer interface, there is now also the option to extend ELAN by recognizer software that runs as a stand-alone executable or script.

The interfacing specification for the second option can be found here:
<http://www.mpi.nl/research/research-projects/language-archiving-technology/avatech/>

DIRECTORY STRUCTURE

The directory structure of this API and Demo package is as follows:

```
recognizer/  
  api/ (api related java source files)  
  lib/ recognizer-api-xxx.jar  
  src/ (demo source tree)  
  docs/recognizer.cmdi  
      doc.html  
      README.pdf  
      example_recognizer.cmdi  
  pom.xml
```

- The api directory contains sources from ELAN that are relevant for an audio-, video- or other recognizer extension
- The lib directory contains the latest recognizer-api.jar, the compiled version of the source files in the api directory. This jar is added to the classpath in the Maven build file
- The src directory contains a package sub directory with the demo recognizer sources in it
- The README is what you are reading now
- The pom.xml file is a Maven build script for compiling and packaging
- The recognizer.cmdi file is a metadata file for the demo recognizer.
- The example_recognizer.cmdi file is an example metadata file for a recognizer extension. A cmdi file is a mandatory part of a recognizer distribution.
- The doc.html is a documentation file about the demo recognizer. This documentation file can be used to provide information to the users about the recognizer via the help option in the recognizer tab. To do so, this file has to be linked through the (optional) documentation element of the recognizer's cmdi file.

BUILD AND DEPLOYMENT INSTRUCTIONS (Recognizer API):

1. Run “mvn compile” and “mvn package” to compile and build the sources
2. Copy the resulting .jar, the demo recognizer.cmdi and the doc.html file to a directory in the extensions directory of ELAN

3. Run ELAN and you will find the Demo Recognizer in the list of recognizers in the Recognizer panel.

DEVELOPER INFORMATION (Recognizer API)

The files `src/main/java/nl/mpi/recognizer/demo/DemoRecognizer.java`, `DemoRecognizerPanel.java`, `docs/recognizer.cmdi` and `docs/doc.html` implement a simple demo recognizer that illustrates the basic communication with ELAN. The only thing this demo recognizer does is creating a segmentation based on a user definable constant interval. By default it does so in the first seconds of the file, but if there are example selections provided by the user, the interval in which to create segments is based on the extent of all selections.

After building and deploying it will present itself as "Demo Recognizer" in the list of recognizers on ELAN's Recognizer Panel. It is advisable to build and deploy this recognizer before implementing one yourself. Its behavior in ELAN combined with the documentation in the source files should give you the information needed to implement a more useful recognition algorithm.

If you have access to the ELAN sources you can also have a look at the implementation of the `SilenceRecognizer` in the `mpi.eudico.client.annotator.recognizer.silence` package.

A recognizer is required to implement the `Recognizer` interface and it can invoke methods of the `RecognizerHost`. The `RecognizerHost` informs the `Recognizer` about the relevant media file(s). The recognizer can present the list to the user, possibly after filtering out the unsupported file types. Before a recognizer's `start()` method is called, the host will call the `validateParameters()` method on the recognizer to verify whether all parameters required to run the recognizer are valid.

While the recognition process runs, the recognizer can give feedback regarding the progress it is making to the `RecognizerHost`. The result of the recognition process must be placed in one or more `Segmentation` objects. They consist of a `MediaDescriptor` that has information about the media that the segmentation refers to and an `ArrayList` with `Segments`.

The `Segments` contain time information and optionally a segment label. The `Segmentation` objects are made available to the ELAN user through the "Create Tier(s)" button. In general a `Segmentation` object translates to a tier and its `Segment` objects to annotations, but ELAN also has an option to customize the conversion from a `Segmentation` to a tier and e.g. create more than one tier from a single `Segmentation` object.

More information on these objects can be found in the Javadoc comments in the following java files in the api directory:

```
from mpi.eudico.client.annotator.recognizer.api
    AbstractSelectionPanel
    ParamPreferences.java
    Recognizer.java
```

RecognizerConfigurationException.java
RecognizerHost.java

from mpi.eudico.client.annotator.recognizer.data

AudioSegment.java
Boundary.java
BoundarySegmentation.java
MediaDescriptor.java
RSelection.java
Segment.java
Segmentation.java
SelectionComparator.java
VideoSegment.java

More descriptions can be found in the Java files. You are NOT supposed to include these classes in your recognizer.jar. They are included in elan.jar and are only included here for documentation purposes.

For this kind of extensions (direct Java plugins) the cmdi file should:

- have the value "direct" for the attribute "recognizerType" of element "recognizer"
- have the fully qualified name of the class that implements Recognizer.java as the value of the attributes "runWin", "runMac" and "runLinux". Only the platforms that are supported need be present.
- have at least one "input" element with value "audio", "video" or "timeseries"

DEVELOPER INFORMATION (Stand-alone component as extension)

Instead of implementing the "Recognizer.java" interface it is also possible to extend ELAN with a stand-alone executable in combination with a proper .cmdi file. You can take the file "example_recognizer.cmdi" file as the basis and modify the xml in accordance with the specifications of your recognizer software.

For this kind of extensions the cmdi file should:

- have the value "local" for the attribute "recognizerType" of element "recognizer"
- have the executable command as the value of the attributes "runWin", "runMac" and "runLinux". Only the platforms that are supported should be present.
- have at least one "input" element with value "audio", "video" or timeseries
- normally have at least one "output" element, e.g. a file for storing the segmentations